## CLAIMS

1. A method of optimising the performance of an interpreter-based runtime system, said runtime system including a virtual machine, the virtual machine adapted to run an application in the context of the runtime environment, the method comprising augmenting the bytecode set of the virtual machine with application-specific opcodes by reference to said application, thereby constituting an application domain-specific virtual machine.

2. A method as claimed in claim 1 wherein the virtual machine is a Java Virtual Machine.

3. A method as claimed in claim 1 or 2 wherein a new domain-specific virtual machine is generated for different categories of applications.

4. A method as claimed in any preceding claim wherein the dynamic and/or static behaviour of the application is used to create new opcode for the domain-specific virtual machine.

5. A method as claimed in any preceding claim wherein the virtual machine is optimized based on the hierarchy of the architecture for which the runtime environment is adapted and/or the semantics of the application which is to be run on it.

6. A method as claimed in any preceding claim wherein the virtual machine is optimized based on a late-binding or dynamic loading model and runtime constant manifestation.

7. A method as claimed in any one of claims 1 to 6 wherein semantically enriched code is statically embedded in the application to enable it to run faster on the virtual machine which is newly generated in accordance with any one of claims 1 to 6.

8. A method as claimed in any one of claims 1 to 6 wherein semantically enriched code is dynamically embedded in the application to enable it to run faster on the virtual machine, said virtual machine newly generated in accordance with any one of claims 1 to 6.

9. A method as claimed in either claim 7 or 8 wherein the semantically enriched code is determined by performing a quantitative trade-off between time and space.

11. A method as claimed in any one of claims 7 to 9 wherein the semantically enriched code is determined based on the dynamic and/or static behaviour of the application.

12. A method of generating an embedded virtual machine for a specific domain of applications based on embedding semantically enriched code in the interpreter loop of the virtual machine.

13. A method as claimed in claim 12 wherein the semantically enriched code embedding step is performed dynamically on newly loaded portions of the application in dynamic languages.

14. A method as claimed in claim 13 wherein the interpreter is dynamically enhanced.

15. A method as claimed in any preceding claim wherein secondary codes are used to accommodate the interpretation of new semantically enriched codes.

16. A method as claimed in claim 15 wherein the encoding of the new semantically enriched codes of the instruction set of the virtual machine is performed for efficient decoding of the most frequently executed codes.

17. A method as claimed in claim 15 or 16 wherein if, a particular code is used very frequently, it is made into a single byte code and the rest of the semantically enriched codes are accommodated by secondary codes.

18. A method of optimising the performance of applications running on an interpreter-based runtime system, the method comprising augmenting the bytecode set of the interpreter with application-specific opcodes by reference to said application, thereby constituting an application domain-specific virtual machine.

19. A method of symbolically executing semantically enriched code opcode or virtual machine instructions for the purpose of integrated code generation and optimization, wherein said method of execution is adapted so as to be based on the semantics of an application which is to be run on the runtime environment thereby increasing the efficacy of

the interaction between the application and the environment in which it is to be executed.

20. A computer systems adapted to perform the method as claimed in any one of claims 1 to 18.

21. A computer program adapted to perform the method as claimed in any one of claims 1 to 18.